PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/46	A2	(11) International Publication Number: WO 96/18149 (43) International Publication Date: 13 June 1996 (13.06.96)

(21) International Application Number: PCT/SE95/01484

(22) International Filing Date: 8 December 1995 (08.12.95)

(30) Priority Data:
9404295-9 9 December 1994 (09.12.94) SE

(71) Applicant *(for all designated States except US)*: TELEFONAKTIEBOLAGET LM ERICSSON (PUBL) [SE/SE]; S-126 25 Stockholm (SE).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: HOLTE-ROST, Anna, Naemi, Ingeborg [SE/SE]; Glanshammarsgatan 213, S-124 72 Bandhagen (SE). ANDERSSON, Sofia, Birgitta [SE/SE]; Vägmästarbacken 1B, S-163 54 Spånga (SE). LARRUY, Ramon, Alexander [SE/SE]; Apelsinvägen 21, S-741 31 Knivsta (SE).

(74) Agents: BJELLMAN, Lennart et al.; Dr. Ludwig Brann Patentbyrå AB, P.O. Box 1344, S-751 43 Uppsala (SE).

(81) Designated States: AU, BR, CA, CN, FI, JP, KR, MX, NO, SE, SG, US, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

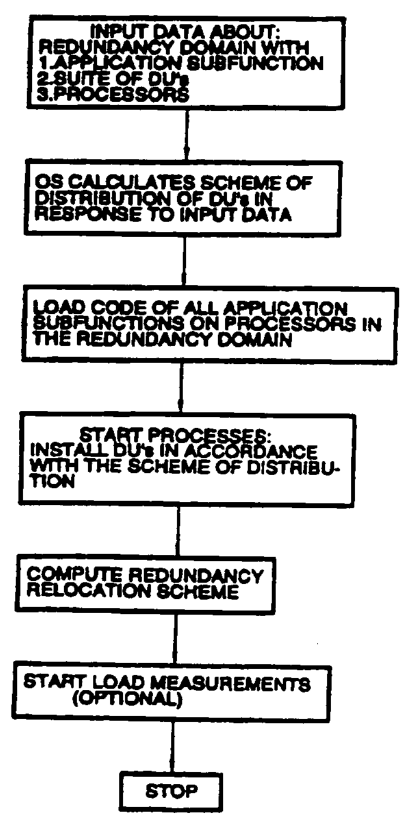
Published
Without international search report and to be republished upon receipt of that report.

(54) Title: CONFIGURATION MECHANISM

(57) Abstract

The present invention relates to a method of configuring a distributed computer system. The method comprises inputting to the system how much processor capacity is required by each application subfunction executing on the system, and how many processors each application subfunction must run on. The processors form a Redundancy Domain. All Resources, grouped together in Distribution Units (D.U.), are distributed over the processors in said Redundancy Domain in proportion to the capacity of the processors and taking into account expected load on the processors, the expected load optionally being based on historical data of processor load. A Redundancy Relocation Scheme is created by assigning at least one other processor within said Redundancy Domain to each D.U., in case of a need to relocate a D.U. to other processors due to load imbalances or other malfunctions in the system.

Initial config.:



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgystan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo			SE	Sweden
CH	Switzerland	KR	Republic of Korea	SI	Slovenia
CI	Côte d'Ivoire	KZ	Kazakhstan	SK	Slovakia
CM	Cameroon	LI	Liechtenstein	SN	Senegal
CN	China	LK	Sri Lanka	TD	Chad
CS	Czechoslovakia	LU	Luxembourg	TG	Togo
CZ	Czech Republic	LV	Latvia	TJ	Tajikistan
DE	Germany	MC	Monaco	TT	Trinidad and Tobago
DK	Denmark	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	US	United States of America
FI	Finland	ML	Mali	UZ	Uzbekistan
FR	France	MN	Mongolia	VN	Viet Nam
GA	Gabon				

CONFIGURATION MECHANISM

Definitions

In the context of this application, the following terms will have the meanings defined below:

"Resource" means something that may be allocated in a distributed system and which one wants to have access to, and therefore must be addressable. Each Resource or Resource Representation has an identifier associated with it, said identifier being common to all objects belonging to a Distribution Unit (see definition in the Detailed Description). It may be subdivided in "Hardware Resources" (such as communication channels, Line Interface Circuits (LIC)), and "Software Resources" (such as subscriber data). Hardware Resources commonly have a software representation, which is in fact what one wants to have access to. Such software representations then become a Software Resource, and constitutes one of the objects that may be handled under the concept of Distribution Units. Thus, for the purpose of this application a "Resource" is a Software Resource.

"Distribution Unit" (D.U.) means a single resource or a group of resources. From a Resource in a D.U. there is a mapping which makes it possible to find the identity of the D.U. In other than telecommunications systems a Distribution Unit might encompass any object in any data base (information about cars, health status of people, etc.), or process allocated objects representing hardware. A "Distribution Unit" in a telecommunications system may e.g. be all subscribers having the same two last digits in their telephone numbers. This creates a total of 100 Distribution Units, although of course each Distribution Unit then represents very many actual subscribers. In a generalized case a Distribution Unit may encompass almost anything, and exactly what it encompasses is dependent on the application in

question. D.U.'s will be defined during system design, and the information about a D.U., i.e. exactly what it encompasses, will be loaded into the system at initial start. D.U.'s may also be added in runtime during operation of the system.

As defined above a Distribution Unit may encompass a number of resources, and what the Distribution Unit actually comprises is dependent on the application in question. Thus it can be almost anything from groups of objects in data bases to process allocated objects representing hardware. In a telecommunications system a very common object making up a Distribution Unit is a group of data representing individual subscribers connected to the telecommunications system.

Each Distribution Unit has a unique identity, that can be used by an application subfunction addressing the Distribution Unit. This identity is composed of one part identifying the application subfunction with which the Distribution Unit in question is associated, and one part identifying the Distribution Unit within this application subfunction.

This may be exemplified with the application being ISDN and the application subfunction being "call access". If the Distribution Unit then is the 12th Distribution Unit within this application subfunction the identity would be ISDN, "call access" 12.

In the case of subscribers in a telecommunication system, the common identifier for objects of an Distribution Unit may be the last two digits of a telephone number, which would create 100 Distribution Units.

Furthermore, each Distribution Unit can be independently allocated to a processor.

The number of Distribution Units defined by an application determines the maximum number of processors that this application might be distributed to. An advantage of the use of D.U.'s as

contemplated in the invention, is that vast amounts of data will become easier to handle, simply because there will be fewer units to handle. Also loss of processors will be handled by the system itself without operator engagement.

Another advantage is an improved addressing capacity, because the length of lists in the addressing mechanisms are dramatically shortened. The "address field" will also be shorter.

"Redundancy Domain" means a specified number of processors on which code for one and the same application subfunction is loaded, such that the subfunction in question may be executed on all processors in the Redundancy Domain. The number of processors within the Domain may be changed during runtime. The "redundancy" resides in the fact that the system will continue to function even if a processor is crashed or blocked.

"Load" means the relative utilization of each processor in the system at any given time.

"Load imbalance" means a deviation from a desired load that may trigger a reconfiguration. Thus, minor imbalances within predefined limits are tolerated.

"Application" means something executing on a distributed computer system, e.g. telecommunications system. Examples of "Applications" are telephone calling systems, ISDN.

"Application Subfunction" means elementary or basic processes together forming an "Application" (e.g. "off-hook" and "call access" functions, code receiving functions etc.). The code of each subfunction may be executed on several processors. Each processor handles different D.U.'s.

Field of the Invention

The present invention relates in one aspect to methods for

configuration and reconfiguration of software resources and of the execution of such resources, in a distributed computer system, in particular a telecommunications system. In another aspect the present invention relates to a telecommunications system having built in functionality for such configuration and reconfiguration.

Background of the Invention

In a distributed computer system, such as a telecommunications system with an enormous number of subscribers, it is impossible to have data for all subscribers and execution and control functions for all subscribers located on one single processor, i.e. a function must have the capability to execute in parallel on several processors. Furthermore, configuration activities involve distribution of the software resources in order to use the processors as efficiently as possible. Such configuration and reconfiguration is a very complex activity in cases where it is difficult to estimate the load (capacity) used by each application resource. Thus, there exists a need for a tool to distribute them over a number of processors.

Even if it were possible to create an initial configuration of the system that could be characterised as "optimal", this condition would not last for very long because the load on the individual processors may change with time due to the application resources executing on the processors, or by loss of processors (e.g. processor crash, replacement of processors for maintenance etc).

Heretofore, in cases where a processor is lost, processes executing on said processor would be halted until the hardware has been repaired or replaced with new hardware. Such operations could take as long as several tens of minutes or even hours, which has become unacceptable in view of the requirements of today in terms of accessibility of the systems in question.

Therefore there has evolved a need for very rapid reconfiguration of such systems, such that the time during which the systems are inoperative may be brought down to the time frame of seconds or at the most a few minutes.

Also for maintenance purposes it would be advantageous if reconfiguration could be rapidly achieved and without operator intervention, especially if the problem causing the need for reconfiguration occurs at a remote location, where it may take a substantial time before an operator can intervene.

Such mechanisms for "run time adjustments" of configurations are dependent on continuous supply of information of the status of the system, in terms of e.g the load balance in the system. This information can be acquired by monitoring the system with respect to certain parameters for a longer period of time. Data from such measurements would notify the change of load of the processors over the time, e.g. that some processors most of the time are more loaded than others. Such information may be used for initiating reconfigurations in the system, either by a maintenance technician or by functionality "built in" in the system. To enable such run time reconfigurations to be made, while fulfilling the requirements concerning non-disturbance of the operation of the system, there must be powerful mechanisms in the operating system supporting such reconfiguration, and applied by the applications.

Prior Art

US-5,165,018 discloses self-configuration of nodes in a distributed message-based operating system. Run-time configuration is achieved by node-based configuration management processes in accordance with information contained in resource definition messages.

Summary of the Invention

The desired mechanisms, as discussed above, are provided by the methods according to the invention, as defined in claims 1, 3, 7, 8 and 11, of distributing the Resources on the processors in the system in the Redundancy Domain, and by the telecommunications switching system according to claim 13, incorporating such functionality.

The computation of the distribution of Resources in this method is supported by the concept of "Distribution Units" (D.U.), which is conceived for the purpose of this invention. The definition of D.U. is given above under "Definitions".

An advantage of the invention is the improved handling. It becomes simpler to configure the system since the distribution of the D.U.'s is supported by the operating system. Since the OS supports load balancing in runtime the requirement of perfect initial configuration becomes less demanding. The need for acute operator action is reduced. Also since the OS supports the Redundancy Relocation at processor loss, a service person does not need to instantly visit the location. The mechanism for Redundancy Relocation is the subject of Swedish patent application no. 95XXXXXX-X, and entitled "Processor Redundancy in a Distributed System" with the same assignee as for the present application, and filed on the same day as the present application.

Still another advantage of the invention is the inherent robustness, i.e. the system will be able to automatically take care of load imbalances due to certain types of anomalities, e.g. processor crash etc. The disturbance on the system is reduced, and the load distribution over remaining processors will be maintained. The built-in Redundancy Relocation reduces the time that a function is unavailable due to e.g. processor loss.

Also the cost will be reduced by virtue of the built-in load balancing, since the processor utilization thereby will be advantageously improved. It may even be possible to reduce the

number of processors in the system by more efficiently distributing Resources over processors.

Brief Description of the Drawings

In the drawings

Fig. 1 is an overview of a telecommunications switching system wherein the present invention may be implemented;

Fig. 1b is a partial view of three processors forming a Redundancy Domain;

Fig 1c shows a Scheme of Distribution of Distribution Units over a Redundancy Domain;

Fig. 2a is a flow chart showing Initial Configuration mechanism in accordance with the invention;

Fig 2b shows a Redundancy Relocation Scheme;

Fig. 3a is a flow chart showing Additional Installation/Removal of Distribution Units in accordance with the invention;

Fig 3b shows the old scheme of distribution together with the new scheme, and the operations performed in order to achieve the new scheme;

Fig. 4a is a flow chart showing (system initiated) Load Balancing in accordance with the invention;

Fig. 4b is a flow chart showing (operator initiated) load balancing in accordance with the invention;

Fig. 4c shows old and new distribution scheme and

operations at load balancing;

Fig. 5a is a flow chart showing manual blocking of processor;

Fig. 5b shows the old scheme of distribution and the Redundancy Relocation Scheme plus the new scheme of distribution. Operations required for achieving the new scheme are also shown;

Fig. 6a shows automatic blocking of a processor;

Fig. 6b shows the old scheme of distribution and the Redundancy Relocation Scheme plus the new scheme of distribution. Operations required for achieving the new scheme are also shown;

Fig. 7a shows processor deblocking;

Fig. 7b shows the old and the new Scheme of Distribution. Operations required for achieving the new scheme are also shown; and

Fig. 7c shows addition of a processor.

Detailed Description of the Invention

The invention will now be exemplified with reference to the drawings and to some different situations wherein the inventive concept may be used. There are two basic situations, namely

A. Initial configuration (when system is started)
B. Reconfiguration (initiated by a technician or initiated by the system itself when the system has been in operation for some time)

Situation B. may in its turn apply to a number of different

situations, such as

a. Additional installation of Distribution Units
b. Removal of Distribution Units
c. Load balancing (manual or automatic)
d. Manual blocking of processor
e. Automatic blocking of a processor (processor crash)
f. Deblocking of processor including return of processor after blocking and addition of processor

There are a couple of different cases of (re)configuration of Distribution Units that this mechanism, together with the prerequisited mechanisms, supports.

The prerequiseted mechanisms are:

- There is a name addressing mechanism in the OS having the capability to address D.U.'s.
- The OS has a "secure" data base, in which configuration data and schemes are stored. They will thus survive a processor crash. Data base is accessible from all processors.
- There is a "state transfer" mechanism for transferring data from "old" to "new" processes.

For each case of (re)configuration two schemes are computed. The first scheme, Scheme 1 for distribution of Distribution Units on the processors within the Redundancy Domain, describes how to distribute the Resources of the system, grouped as Distribution Units (as defined previously herein) over the processors in the system, taking into account parameters determining the capacity of each processor, optionally measured values of load.

The other scheme, Scheme 2, for redundancy relocation describes how the Distribution Units will be moved in case of loss of a

processor because of blocking.

This scheme will outpoint for each Distribution Unit indicating to which processor it will be relocated in the case of a reconfiguration at a later stage, e.g. in situations a - f above.

Now we will deal with the basic situations with reference to an Example, and the Figures wherein the distribution of D.U.'s over the processors in a Domain is shown. Arrows show the Redundancy Scheme. Italics show the Scheme to be attained after reconfiguration.

EXAMPLE:

In the drawing figures, wherein i.a. flow charts of respective case and corresponding relocation scheme are shown, the suites marked with italics indicate a recalculated scheme of distribution to be attained after the reconfiguration operations have been invoked.

In Fig. 1a there is shown a simplified overview of a telecommunications system. It comprises several subscribers, each having at least one telephone 1. Each telephone is connected to the system via a so called LIC (Line Interface Circuit), which in its turn is connected to a switch 2 responsible for the communication between all units in the system. On all processors 3 in the system there are one or more subfunctions executing. There is also provided a data base 4 containing subscriber data etc. The data base is part of the operating system executing on the processors, and normally resides in primary memory (some of the processors may also be equipped with a disk, that may contain the data base). The disks are otherwise primarily used as loading media and for back up purposes, but may also have other functions.

In Fig. 1b there is shown three processors 3 forming a Redundancy Domain (indicated in broken lines), as defined previously, in the

system of Fig. 1a. Each processor is loaded with code for an application subfunction APPL and an operating system OS, comprising the data base.

Let us assume that this "sub"-system is to be configured with an application. An application subfunction will have 30 Distribution Units executing. They are numbered in a sequence (0-29) (a suite).

The dimensioning has resulted in a Redundancy Domain consisting of the three processors shown in Fig. 1c.

In this simplified example, we assume that the processors are all equal, and that no other applications will be executing on the processors within the Redundancy Domain, i.e. the Distribution Units will (initially) be equally distributed over the processors within the Redundancy Domain.

A. Initial configuration

In setting up a telecommunications system it must of course be dimensioned to the expected utilization, but this is a matter of standard procedures and does not form part of the invention and will hence not be discussed here. Thus, when the dimensioning of the system is done it has to be decided how much capacity will be needed by each application subfunction that is going to execute on the system. In this process one also has to consider cases of loss of execution capacity, e.g. due to processor crashes or other losses of processors, such as removal of one processor for maintenance purposes etc. This means that it has to be determined how many processors the system should comprise.

Thereafter it has to be decided how many processors that each application will execute on, i.e. to create an initial configuration of "Redundancy Domains". Thus, for each application one determines a number of processors on which the same code will be loaded, and this group of processors constitutes a Redundancy

Domain for each application subfunction in question. A consequence of this is that all processors within each Redundancy Domain will be able to execute the application subfunction associated with this Redundancy Domain. The code of the application subfunction will thus be loaded on all processors within the domain.

When a Redundancy Domain is initially configured the Distribution Units within the application are allocated to the Redundancy Domain. It is the (re)configuration mechanism (operating according to a distribution function

$$D = f(P0...Pn, L0...Ln, DU0...x)$$

wherein P=processor, L=load, n=processor designation, DUx= the x:th Distribution Unit in a Redundancy Domain) that decides how the Distribution Units will be distributed over the processors within the domain based on the input data, such as number of processors, processor capacity, processor speed, load balance, number of Distribution Units etc.

In our example application subfunction (Fig. 1c) there has been defined 30 Distribution Units with identifiers in a sequence from 0-29. In this example they are divided into three groups (equally distributed), because there are three equal processors within the defined Redundancy Domain.

When the scheme of distribution of the Distribution Units amongst the processors (Figure 2b) is initially calculated, also the scheme of redundancy relocation of Distribution Units is calculated, so that each Distribution Unit has a redundant processor within the domain, outpointed.

The scheme is calculated in order to keep the suites of Distribution Units in a consecutive order to the greatest extent possible.

Fig. 2a shows a simple flow chart of an Initial Configuration.

In the first step, during input of data (normally data is written to a file which is then read by the O.S.), the operating system is provided with information about which application subfunction(s) the configuration is to handle; how the Redundancy Domain is defined; and the suite of Distribution Units that are to be distributed in the configuration. In response to said data, the operating system will calculate the Scheme of Distribution of DU's over the active processors in the Redundancy Domain (Fig. 1c).

Then the code for the application subfunction(s) will be loaded on the processors in the Redundancy Domain in question, and subsequently the DU's will be installed in accordance with the previously calculated Distribution Scheme, whereby the application Distribution Units are distributed over (and started on) the processors in the domain.

The computation of said Scheme of Distribution for a given application is supported by the concept of Distribution Units, defined previously. To this end a more or less complex algorithm may be used.

The simplest algorithm is to distribute the Distribution Units as equally as possible over the active processors within the Redundancy Domain. However, the algorithm may also already initially consider the memory capacity of the processor, the execution capacity of the processor, and the capacity used by other applications executing on the processors. Thus, the number of Distribution Units allocated to individual processors will be weighted in proportion to such considerations.

Finally a Redundancy Relocation Scheme is calculated.

The following figure (Figure 2b) shows how the Distribution Units allocated to a processor, will be divided into suites, that in case of blocking of a processor will be moved to the most suitable processor still in operation. The arrows in the figure

indicates how DU's will be moved from a processor to another in case of processor loss.

Furthermore, the distribution algorithm preferably maintains the identities of Distribution Units allocated to a processor in sequence, such that when Distribution Units are relocated to another processor during reconfiguration, they will be relocated to maintain the identities of Distribution Units on each processor in sequence to the greatest extent possible (discussed and exemplified further below). In this way the addressing will be much more rapid, because providing sequences in the address tables yield fewer lines in the tables.

The distribution of the Distribution Units (0..29) amongst the processors, has resulted in a suite of 10 Distribution Units per processor (0..9), (10..19), (20..29). Each one of the suites (0..9), ... will be divided into two equally large parts, that will be separately moved to the two remaining processors, in case of a processor blocking.

In a case where there are inherent inequality amongst the processors, then of course the subdivision for the purpose of relocation must take this into consideration, such that perhaps twice as many Distribution Units are relocated to processor P1 than to P2 if P1 has twice the capacity of P2.

The scheme of automatic redundancy relocation will be recalculated after performance of a redundancy relocation, and after each kind of other reconfiguration affecting the distribution.

For operation and maintenance of the application subfunction, an interface is required. Such an interface will have facilities for initial installation of Distribution Units (INSTALL), removal (REMOVE) of Distribution Units, and for moving Distribution Units from one application subfunction executing on one processor to another subfunction executing on another processor (the latter is the subject matter of the copending Swedish patent application

Serial No. 9503339-5, "Synchronisation allowing states transfer during smooth system upgrade" incorporated herein by reference). The functions or commands defined in the referenced copending application are SHUTDOWN and TAKEOVER. Said functions are basically implemented as programs triggering certain activities, and are briefly described below.

SHUTDOWN prepares for termination and a forthcoming take over. TAKEOVER transfers control of resource objects in an old static process to a new static process.

B Reconfiguration cases

Reconfiguration may be made either manually by an operator or automatically by the system itself, such as when certain limit values in e.g. load are exceeded. Manual reconfiguration is extremely complex, and therefore the methods disclosed herein are implemented in the operating system.

To this end there may be continuous measurements of the load balance of the processors in the system, i.e. in order to detect whether one or more processor are more heavily loaded than others (such measurement mechanisms are available in operating systems and do not per se form part of the present invention. Therefore they will not be discussed herein). The result of the measurements are considered and then included in the calculation of a new distribution for the purpose of such reconfiguration. This calls for predefined thresholds for acceptable load, deviation from which would trigger a reconfiguration.

The activities that may be initiated by a maintenance technician in run time are the following:

- Installation of additional Distribution Units. The Distribution Units are allocated to the Redundancy Domain.

- Removal of Distribution Units. The Distribution Units are

removed from a Redundancy Domain.

- Load Balancing. An operator initiated reconfiguration because of bad load balance.

- Manual blocking of a processor. A processor is by some reason taken out of service. All Distribution Units executing on this processor must be moved to other processors within the Redundancy Domain.

- Deblocking (Manual) of a processor. A processor within the Redundancy Domain is taken into service. The Distribution Units are moved in order to take the processor into service and to balance the load.

The activities initiated by the system itself are the following:

- Load Balancing. Long time measurements notifying bad load balance. Thresholds defining when an imbalance is reached are defined. The Distribution Units are moved in order to achieve a better load balance.

- Automatic blocking: Reconfiguration of Distribution Units initiated by a processor crash.

- Automatic deblocking of a processor. A processor within the Redundancy Domain is taken into service. The Distribution Units are moved in order to take the processor into service and to balance the load.

Additional installation of Distribution Units

Additional Distribution Units are added to the Redundancy Domain (Fig. 3a) (an example could be an increased number of subscribers in a telecommunications system). This will cause a new scheme of distribution (and redundancy relocation). In Fig. 3b there is shown an additional installation of the Distribution Units (30-

39), applied to our example.

Thus, the first thing to happen is that the system must be "informed" that 10 new Distribution Units (30-39) will be added. As previously described data is written to a file which is read by the operating system. The scheme of distribution of the Distribution Units amongst the processors within the domain will then be recalculated (0..12), (13..26), (27..29) by the operating system in response to input data. In order to attain the recalculated scheme, the additional (new) Distribution Units are INSTALLED and the other (original) Distribution Units are moved to processors according to the new Scheme of Distribution (with "SHUTDOWN" towards the "old" processes, and "TAKEOVER" against the "new" processes. In the case of removal of old DU's of course no INSTALL but REMOVE is required. When the redistribution has been carried out a new Redundancy Relocation Scheme is (automatically) calculated based on the data representing the system (number of processors, processor capacity etc).

Load Balancing

Let us assume that Processor 2 is overloaded (as an example measurements may indicate that there is a 30% overload, and the operating system is notified accordingly), so four of the allocated Distribution Units should be relocated from processor 2. The Scheme of Distribution (see Fig. 4c) of the Distribution Units amongst the processors within the domain will first be recalculated (0..11), (12..17), (18..29). Then, in order to attain the recalculated scheme the Distribution Units are moved (with "SHUTDOWN" towards the "old" process, and "TAKEOVER" against the "new" processes). When distribution is completed, a new Redundancy Relocation Scheme will be calculated.

Manual blocking of a processor

When a processor is to be manually blocked, e.g. it is to be

removed for maintenance, we want to move all the Distribution Units executing in that processor as quickly as possible. In this case the scheme for redundancy relocation calculated initially will be used. To attain the scheme for automatic redundancy relocation, the Distribution Units allocated to processor 2 are moved to processor 1 (10..14) and processor 3 (15..19) with "SHUTDOWN" towards the "old" process, and "TAKEOVER" against the "new" processes (fig. 5a and 5b).

Automatic blocking of a processor (processor crash)

When a processor is automatically blocked, e.g. due to a processor crash, all process allocated data is lost so no state transfer between the involved processes are conceivable. Accordingly the Distribution Units allocated to the blocked processor will not be moved with "SHUTDOWN-TAKEOVER", but will be "reinstalled" in the processes taking over execution.

The operating system detects a crash by mechanisms known per se. When a crash is detected immediately the DU's allocated to the lost processor will be INSTALLED in accordance with the Redundancy Relocation Scheme calculated at Initial Configuration, or with the last updated Scheme in the case that some event has triggered a recalculation for other reasons. In a more complex situation, additionally some DU's on the still active processors may have to be moved to meet e.g. the load balance criteria.

Fig. 6b shows an example of an automatic relocation of Distribution Units, caused by an automatic blocking of a processor (processor crash).

To attain the scheme for automatic redundancy relocation, the Distribution Units allocated to processor 2 are moved so as to be allocated to processor 1 (DU's 10..14) and processor 3 (DU's 15..19) with "INSTALL" against the "new" processes.

Addition of processor

The process of adding a new processor is shown in Fig. 7c. In the first step the system must be informed similarly to the case of Initial Configuration, i.e. what application subfunctions are used, and what the Redundancy Domain is. Then the new processor is loaded with code for the application subfunctions in question, and finally the processor is blocked.

Return of processor after blocking (deblocking) including addition of processor

When a processor returns from blocking, the Scheme of Distribution of the Distribution Units amongst the active processors within the Redundancy Domain is recalculated. Fig 7a is a flowchart showing the sequence of steps occurring when a processor is taken into operation again after having been blocked. First the operating system detects that a processor has been added or has been reinstalled (both cases are identical from the operating systems point of view. The way detection is carried out is irrelevant, and could be effected by an operator action or by software checking the available slots whether a processor is present or not. The skilled man will appreciate that there are many ways of achieving this function. As soon as the presence of a new or reinstalled processor is detected, the operating system calculates a new Scheme of Distribution including the deblocked processor.

When the new Scheme is calculated, the system will start moving the DU's in accordance with said Scheme.

Fig. 7b shows how the Distribution Units (10..14) and (15..19), are moved between the involved processors (with "shutdown" towards the "old" process and "takeover" against the "new" processes) in order to attain the new scheme of distribution.

CLAIMS:

1. A method of configuration of a distributed computer system, the method comprising

a) distributing all Resources, grouped together in Distribution Units (D.U.) over the processors in a Redundancy Domain in proportion to the capacity of the processors, taking into account expected load on the processors, the expected load optionally being based on historical data of processor load;

b) creating a Redundancy Relocation Scheme by assigning at least one other processor within said Redundancy Domain to each D.U., in case of a need to relocate a D.U. to other processors due to load imbalances or other malfunctions in the system.

2. The method as claimed in claim 1, comprising an initial determination of how much processor capacity is required by each application subfunction executing on the system; and determining the number of processors each application subfunction must run on, said number of processors forming a Redundancy Domain.

3. The method of claim 1 or 2, wherein said D.U.'s are numbered consecutively and wherein the assigning of another processor to said DU is made such that as far as possible the DU's will be ordered in sequence on the other processors.

4. A method of dynamically achieving a more efficient utilization of processor capacity in a distributed computer system, wherein code for an application subfunction is loaded on several or all processors in said system, the processors on which a subfunction is executing thereby forming a Redundancy Domain, comprising the steps of

a) distributing groups of Resources (Distribution Units) according to a Scheme of Distribution such that the processor load is reasonably balanced;

b) continuously monitoring the state of each processor during operation to detect changes in load causing load imbalance among said processors;

c) in response to a detected imbalance, calculating a new Scheme of Distribution for said Distribution Units over said processors, said calculated distribution being weighted in proportion to said change in load balance of said processors;

d) relocating DU's from a heavily loaded processor to other less loaded processors in accordance with said calculated new distribution; and

e) transferring processing from said heavily loaded processor to said less heavily loaded processor(s).

5. The method as claimed in claim 4, wherein said step of relocating DU's involves copying DU's from said heavily loaded processor to a less loaded one while the original data is still in operation on said heavily loaded processor.

6. The method as claimed in claim 5, wherein said Distribution Units are numbered consecutively and originally distributed in sequence on each processor, and wherein the calculated new distribution relocates Distribution Units to other processors such that the number sequences will be continuous to the greatest extent possible.

7. The method as claimed in any preceding claim, wherein changes in load are only considered if they exceed predefined limits.

8. A method of relocation of resources executing on a processor in a distributed computer system to another processor or other processors, said resources being grouped together in Distribution Units, and said Distribution Units being distributed among said processors in said system in accordance with parameters determining the capacity of each processor in said system, the method comprising the following steps:

a) monitoring the state of each processor during operation, and detecting changes caused by processor loss;

b) installing Distribution units associated with the lost processor on a new processor according to a Redundancy Relocation Scheme;

c) calculating a new Redundancy Relocation Scheme.

9. A method of distribution of software resources executing simultaneously on a plurality of processors in a distributed computer system, said processors forming a Redundancy Domain, comprising grouping software resources into Distribution Units, and distributing groups of such Distribution Units over the processors in said Redundancy Domain according to a scheme of distribution derived and calculated on the basis of parameters determining the capacity of each processor in said system.

10. The method of claim 9, wherein the distribution is calculated by evenly distributing said Distribution Units amongst the processors.

11. The method of claim 9, wherein the distribution is calculated by weighting the number of Distribution Units associated with a processor such that fewer DU's are associated with processors with heavy load, as detected in step a), or with lower inherent capacity.

12. A method of achieving a more efficient utilization of processors in a distributed computer system, comprising

a) calculating a scheme of distribution of Distribution Units over said processors, taking into account parameters determining the capacity of each processor, such that the calculated distribution is weighted in proportion to differences in capacity of said processors;

b) distributing groups of Resources (Distribution Units) over a selected number of processors (forming a Redundancy Domain) in said system, such that the processor load is reasonably balanced, according to said scheme of distribution;

c) loading code for an application subfunction on the processors of each Redundancy Domain in the system, such that the application subfunction may be executed on all of the selected processors;

d) installing Distribution Units in accordance with said Scheme of Distribution; and

e) running the system.

13. A method as claimed in any preceding claim, wherein the distributed computer system is a telecommunications switching system.

14. A telecommunications switching system comprising a plurality of processors, each having an operating system which comprises functionality for carrying out the methods as claimed in the preceding claims.

15. A method of configuration of a distributed computer system, in particular a distributed telecommunications system, the method

comprising

determining how much capacity is needed by each application subfunction, taking into consideration loss of capacity due to e.g. processor crash;

determining how many processors that each application subfunction will execute on, said processors forming a Redundancy Domain, such that all processors within this domain will be able to execute the application subfunction in question, whereby code for the application subfunction will be loaded on all said processors; and

distributing the Resources to application processes executing on these processors.

Fig.1a

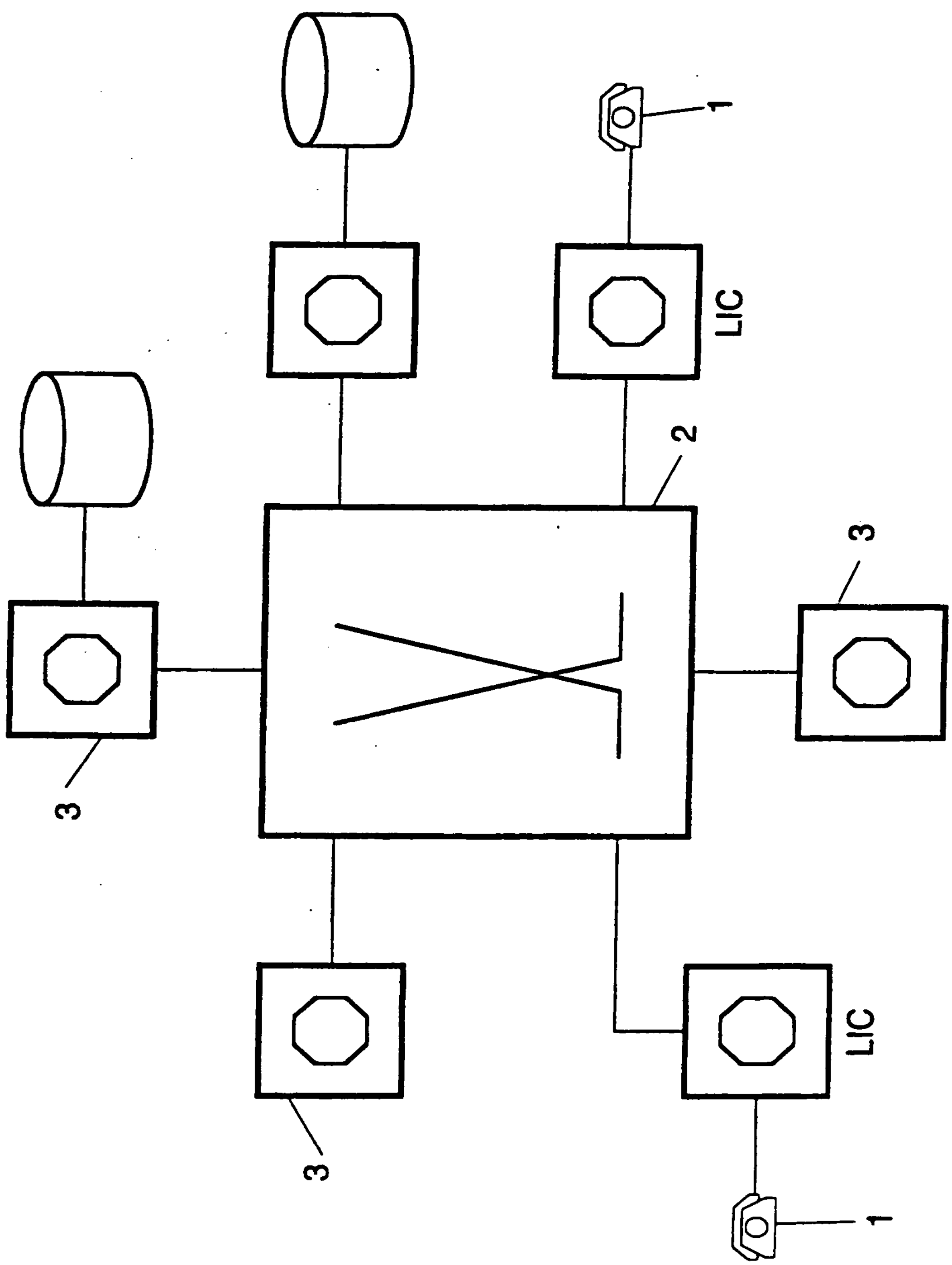


Fig.1b

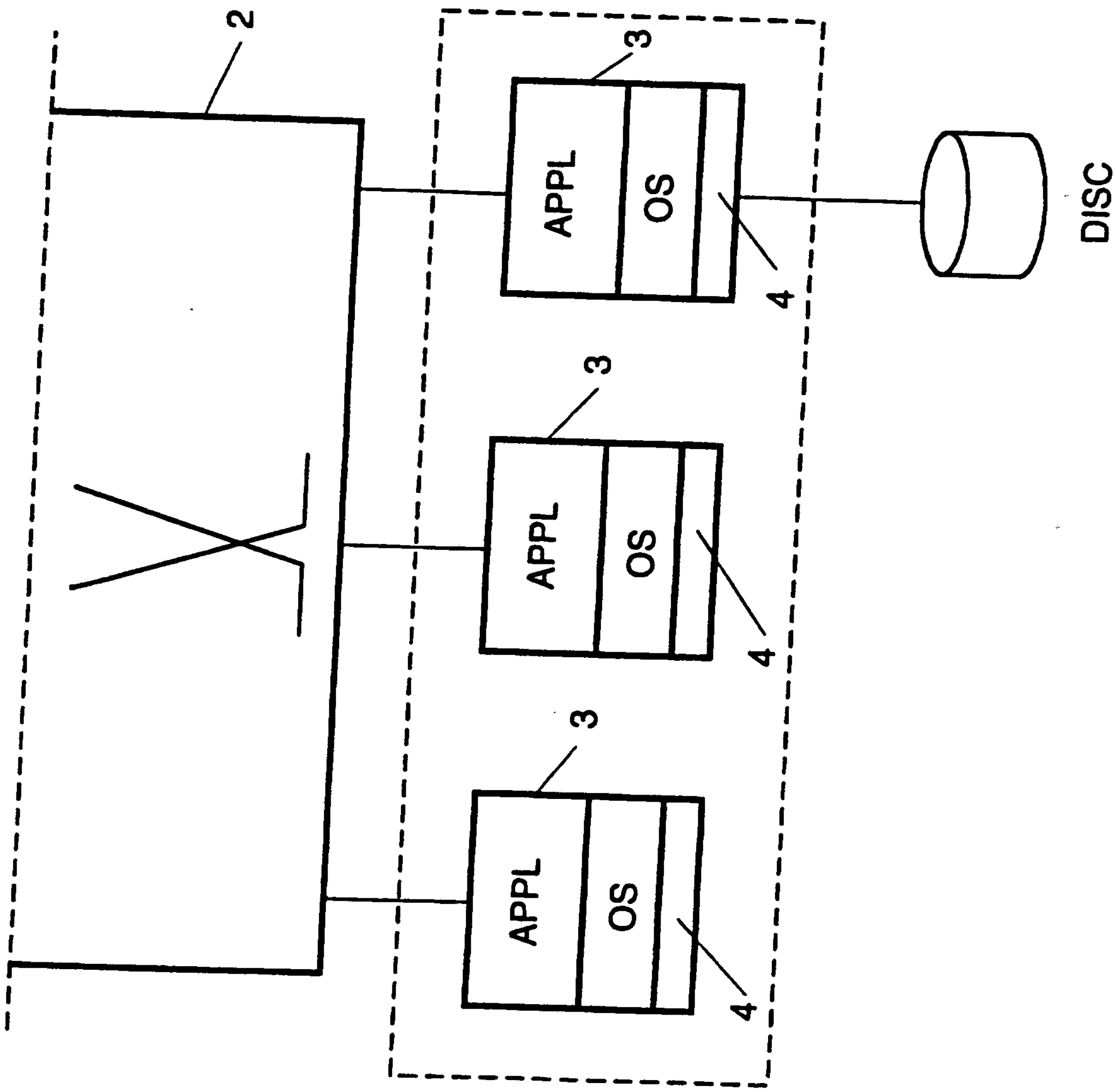


Fig.1c

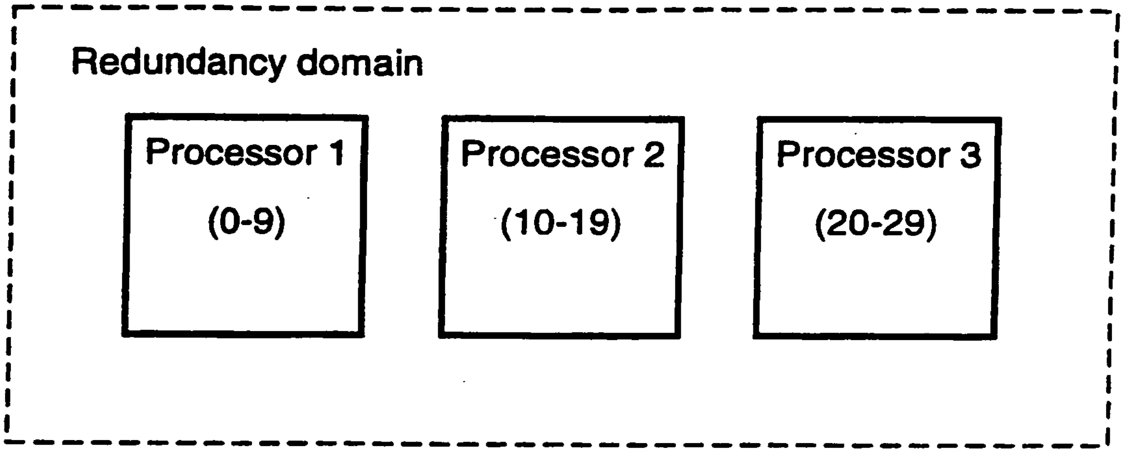


Fig.2b

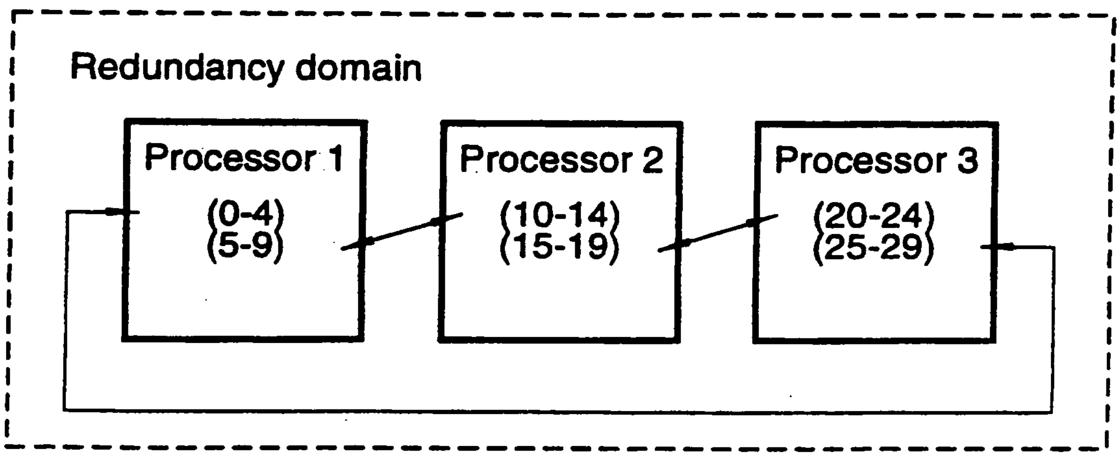


Fig.3b

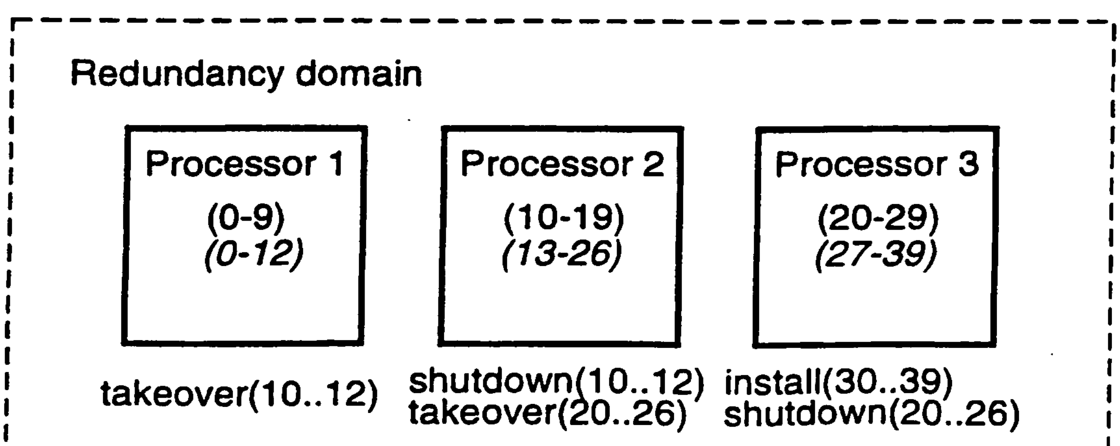


Fig.2a

Initial config.:

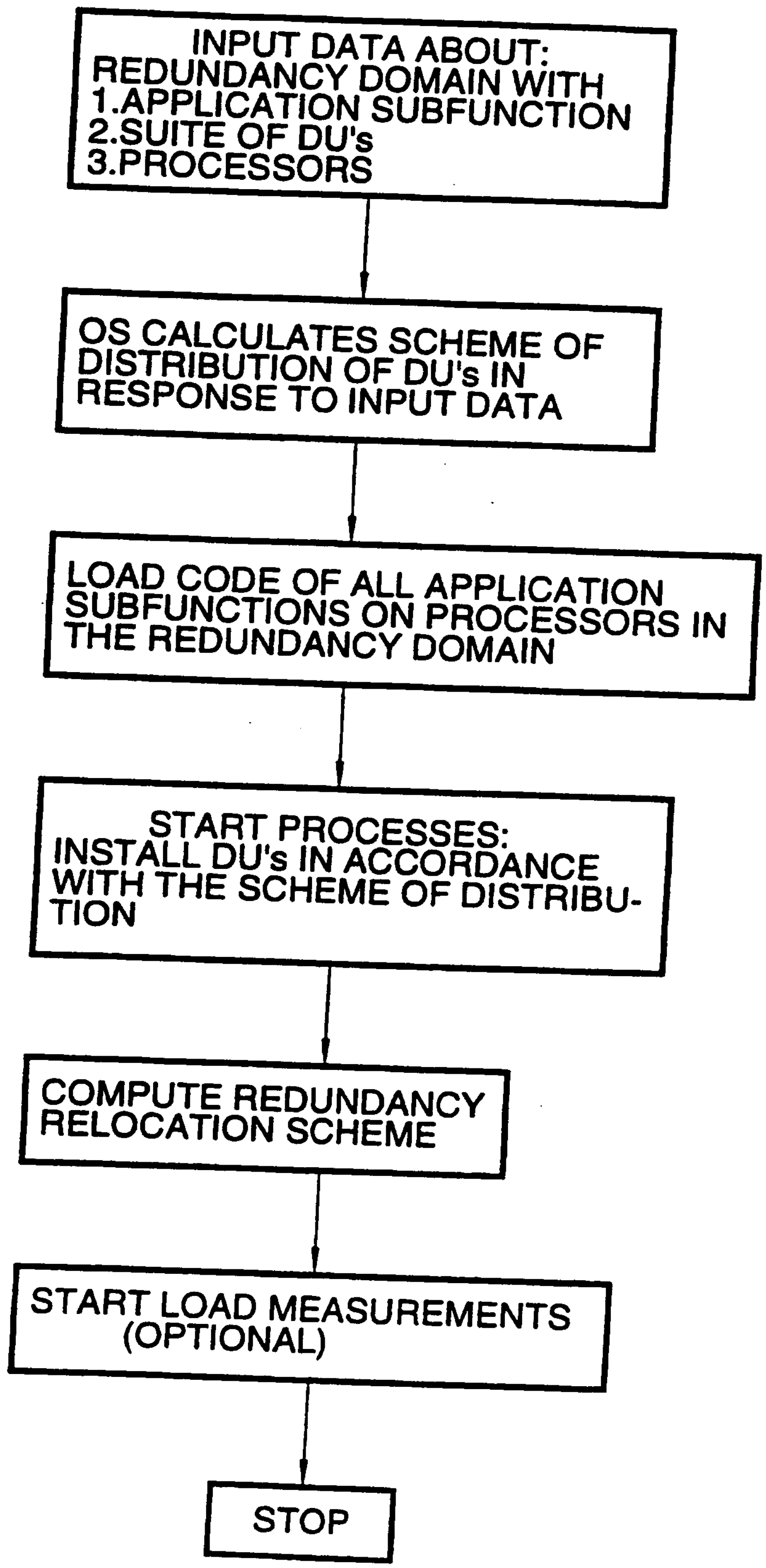


Fig.3a

Add DU/REMOVE DU:

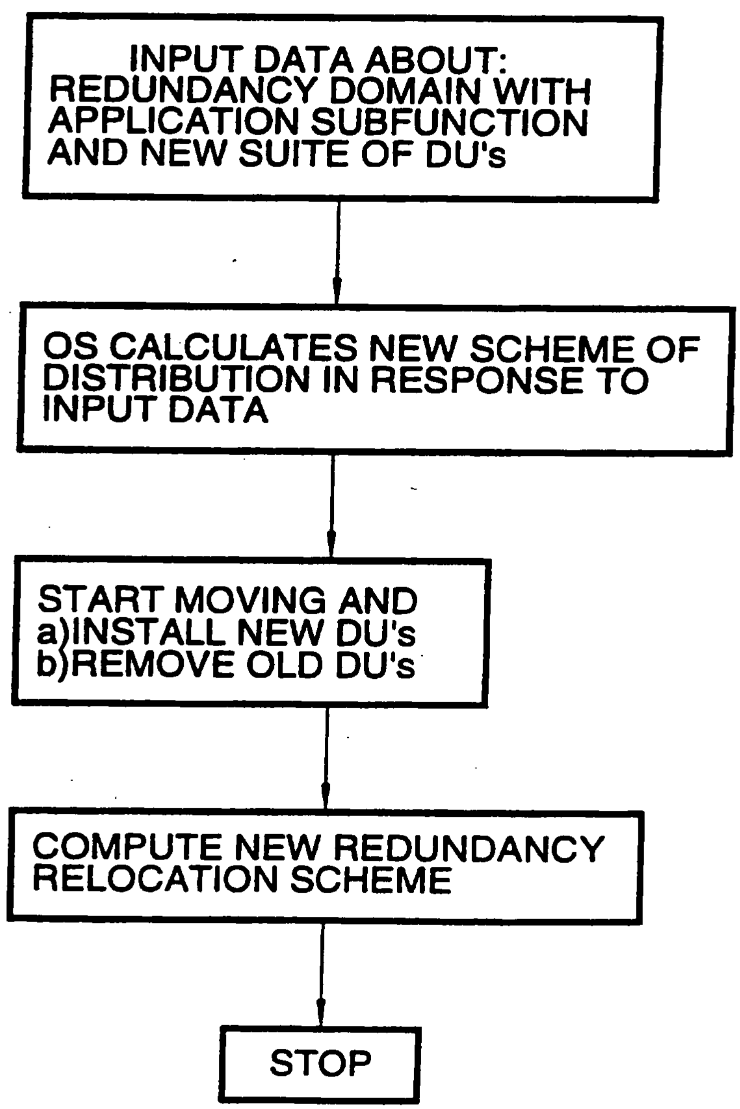


Fig.4a

Load Balancing:

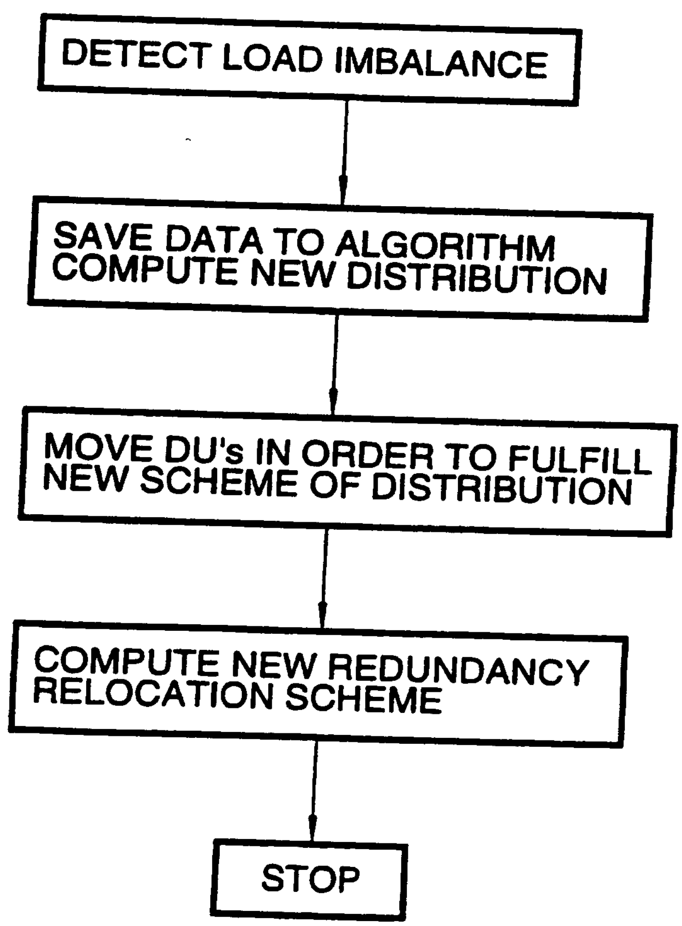


Fig.4b

Manual Load Balancing:

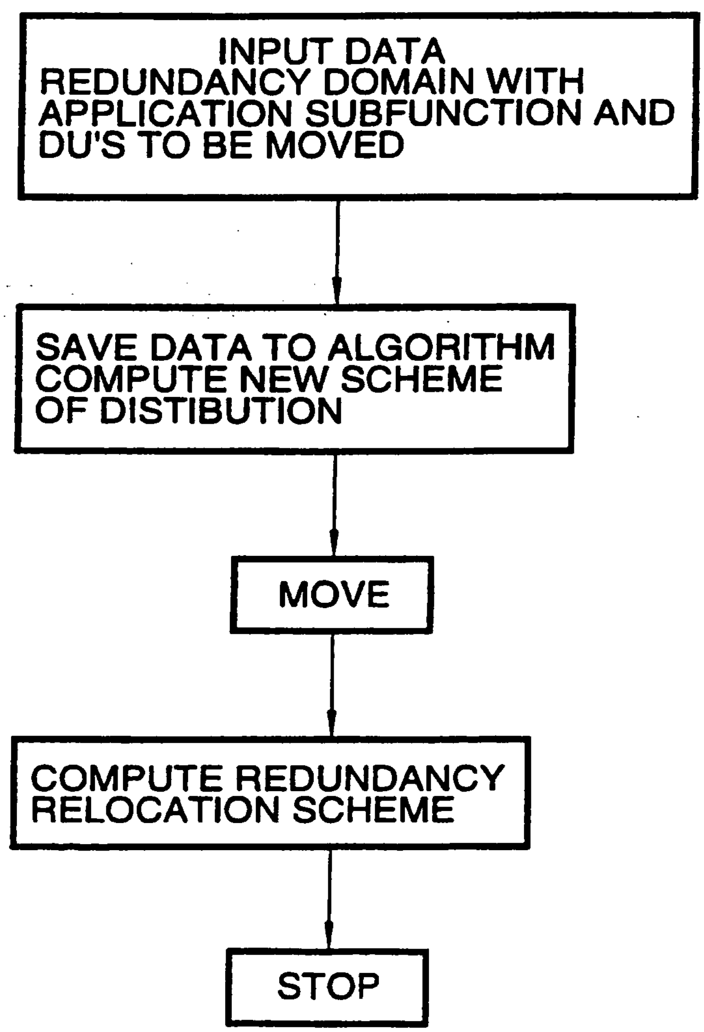


Fig.5a

Manual Blocking:

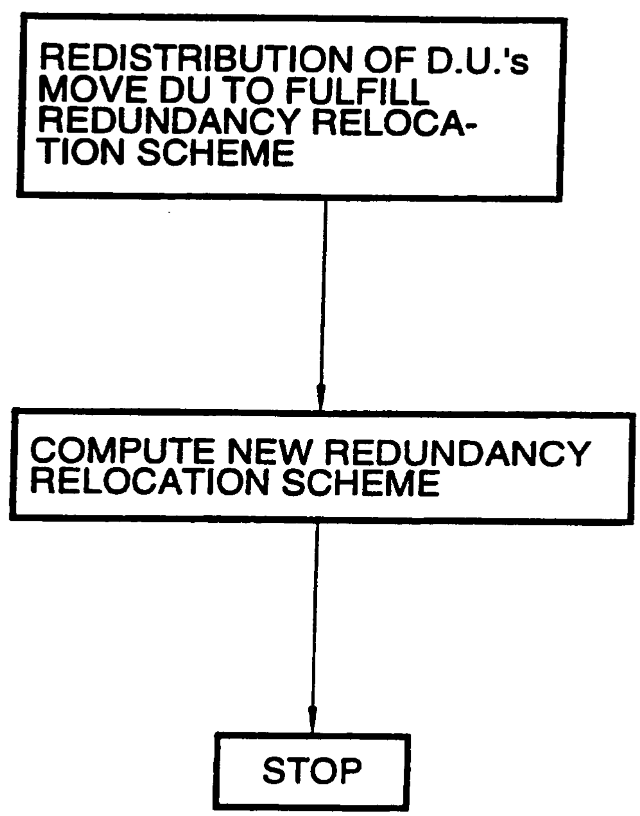


Fig.4c

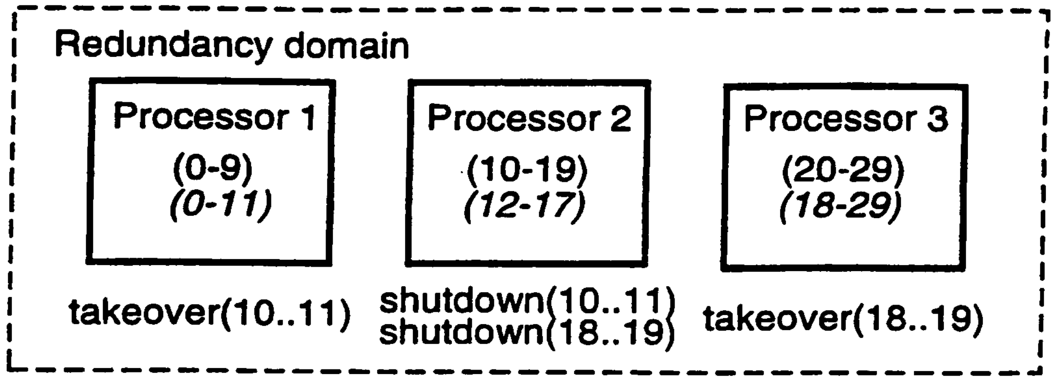


Fig.5b

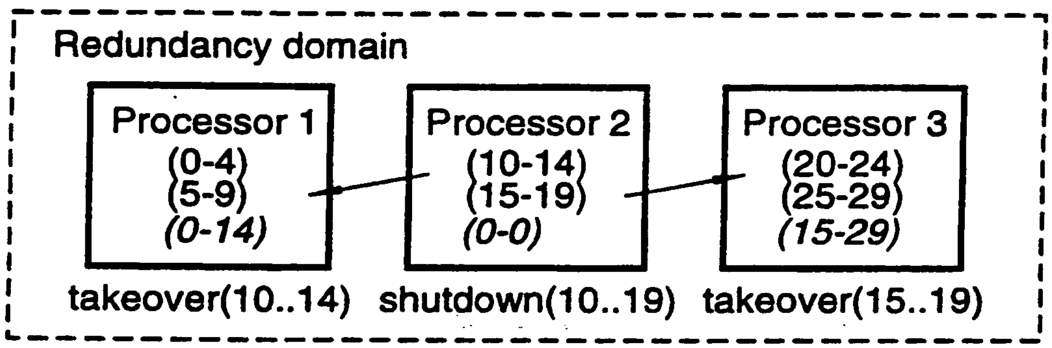


Fig.6b

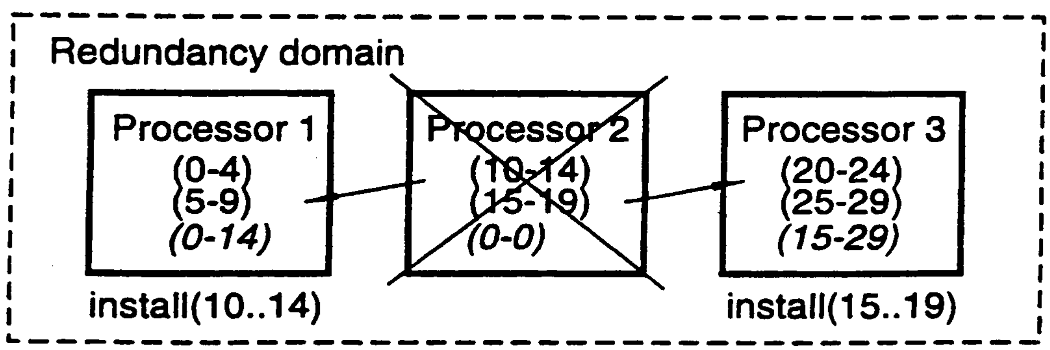


Fig.7b

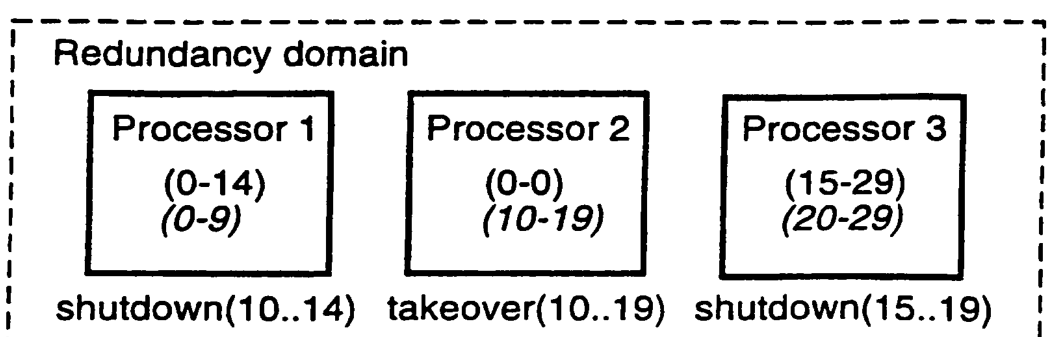


Fig.6a

Automatic Blocking (Processor crash):

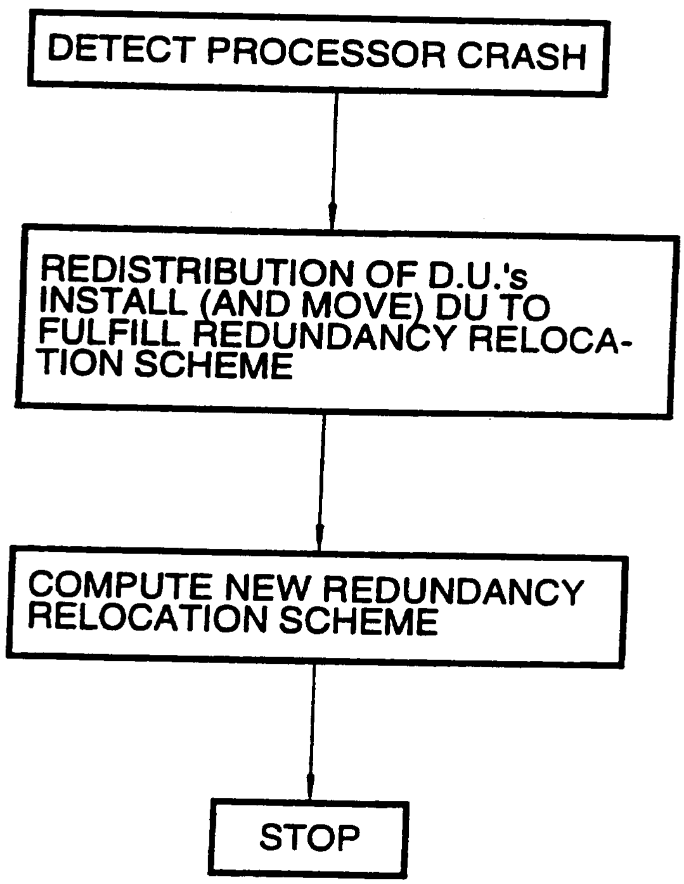


Fig.7a

Processor Deblocking:

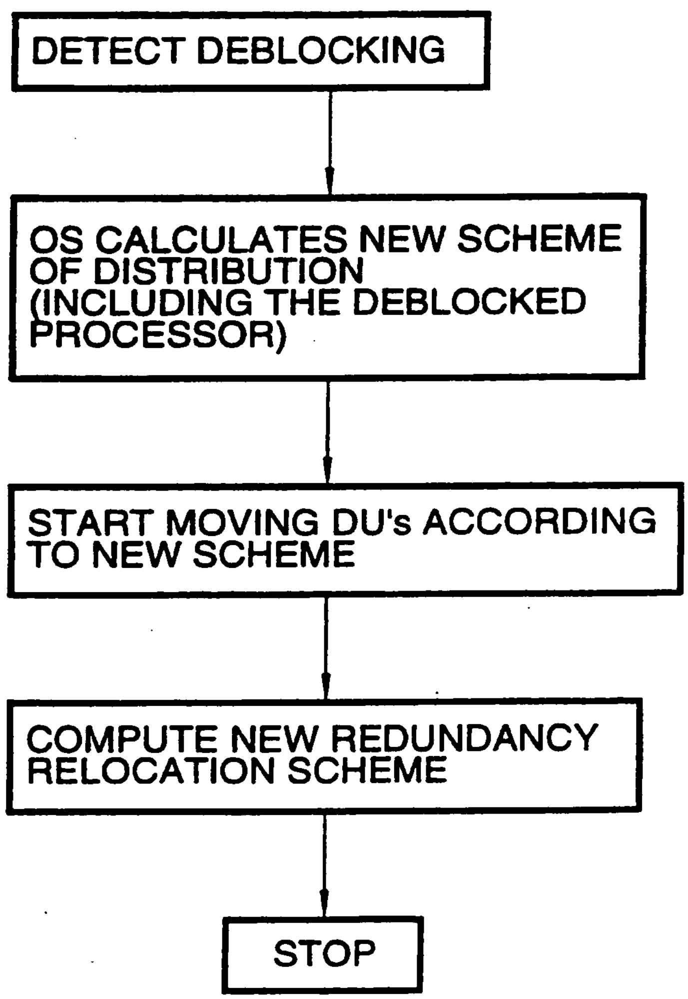


Fig.7c

Add Processor: